REMARKS

Claims 1-6 and 23-30 were presented for examination. The Office Action dated September 29, 2008 rejects all pending claims.

This paper amends claims 1, 4-6, and 23-27, and adds claim 31. Support for the amendment can be found at least in paragraphs [0033] and [0039] of the applicant's specification. Applicant is not conceding that the subject matter encompassed by claims 1, 4-6, and 23-27 prior to this Amendment is not patentable over the art cited by the Examiner. Claims 1, 4-6, and 23-27 were amended in this Amendment solely to facilitate expeditious prosecution of the application. Applicant respectfully reserves the right to pursue claims as presented prior to this Amendment, including the subject matter encompassed by claims 1, 4-6, and 23-27 and additional claims in one or more continuing applications.

Claims 1-6 and 23-31 are now pending in the application.

35 U.S.C. § 112

Claims 24-26 rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant has amended claims 24-26 to remove the indefiniteness described by the Office Action and submits that the rejection is overcome.

35 U.S.C. § 102

Claims 1-6 and 23-30 are rejected under 35 U.S.C. 102(e) as being clearly anticipated by Baber et al in US Patent No. 7,171,691 (hereinafter Baber). Applicant respectfully traverses the rejection to the extent it is maintained against the claims as amended, because Baber does not teach or suggest every element and limitation of the applicant's invention as now claimed.

A claimed element of the applicant's invention, as set forth in independent claim 1, is the editable configuration file that includes the definitions of potentially harmful active content. According to the applicant's specification, as new potentially harmful active content becomes known, an administrator can edit the configuration file to include new definitions, thus keeping pace with the development of the new potentially harmful active content (paragraph 16). Upgrades to software are thus not required to change the filtering behavior of applicant's active content filter. Editing this configuration file is what changes (i.e., configures) filtering behavior (paragraph 28).

To use this editable configuration file in order to identify potentially harmful active content in a document, the applicant's claimed invention includes "parsing the editable configuration file to generate a data structure representation of the one or more definitions in the editable configuration file" and "comparing a data structure representation of the document with the data structure representation of the one or more definitions of potentially harmful active content."

Baber teaches a content sanitizing transcoding engine that includes patterns for identifying malicious content in a document. The patterns used to find malicious content are XPath expressions or XSL transformation patterns (col. 4, ll. 19-20). Once the document is parsed and formed in a document object model, the XPath or XSL transformation patterns are used to locate virulent patterns in within the XML document (col. 4, ll. 24-27). Node elements of the document that contain the potentially malicious content are annotated and subsequently sanitized.

According to Wikipedia (web page attached), XPath (XML Path Language) is a language for selecting nodes from an XML document. An expression in Xpath defines a navigation path through a document tree, such as a tree representation of the XML document. The path expression can select nodes in the document according to specified criteria. Xpath is implemented and run through API from languages, such as Java, C# or JavaScript, or embedded in languages, such as XSLT. Execution causes navigation of the document along the path described by the path expression. The result of executing an Xpath or an XSLT expression is to select a node in the document. Thus, an Xpath expression is generally code executed by other code in order to select a node in the document.

Baber, however, does not teach putting the new Xpath expression into an editable configuration file with other Xpath expressions or generating a data structure representation of a configuration file. Hence, Baber cannot be seen to disclose or suggest providing one or more definitions of potentially harmful active content into an editable configuration file, parsing this configuration file to generate a data structure representation of the one or more definitions, and comparing this data structure representation with the data structure representation of the document, as now set forth in the claims.

The differences of the applicant's invention from Baber become more apparent when compared in the light of a newly developed (e.g., previously unknown) potentially harmful active content. Users of the applicant's invention adapt by adding a new definition, identifying the new potentially harmful active content, to the editable configuration file; whereas users of Baber may adapt by generating a new Xpath expression. In the case of the applicant's invention, adding the new definition is sufficient to put the new definition into effect -- code does not need to be modified -- the configuration file is parsed, a data structure representation of the definitions is generated, and this data structure representation is compared with that of the document. In Baber, however, generating a new Xpath expression may not be enough to put the new Xpath expression into effect. Code would need modifying to call that new Xpath expression (e.g., through an API) or to embed that new Xpath expression into another language, such as XSLT.

For a reference to anticipate the applicant's invention, that reference must show each element and limitation of the applicant's claims. As noted above, Baber does not teach or suggest the applicant's claimed editable configuration file, parsing this editable configuration file to generate a data structure representation of the one or more definitions in the editable configuration file, and comparing the data structure representation of the document with the data structure representation of the one or more definitions of potentially harmful active content to identify potentially harmful active content within the document, as now set forth in the applicant's claimed invention. Therefore, applicant respectfully submits that the amendment to the claim overcomes the rejection and requests that the rejection be withdrawn.

Each dependent claim depends directly or indirectly from patentable independent claim 1, and incorporates all of its limitations and, therefore, is patentably distinguishable over the cited reference for at least this reason. Moreover, each dependent claim also recites an additional limitation, which, in combination with the elements and limitations of its independent claim, may further distinguish that dependent claim from the cited reference. Therefore, applicant respectfully requests the withdrawal of the rejection of these claims.

New claims 31

Claims 31 is a dependent claim, which depends from patentable independent claim 1, and therefore is allowable as written for at least this reason. Moreover, the dependent claim also recites an additional limitation, which, in combination with the elements and limitations of its independent claim, may further distinguish that dependent claim from the cited reference.

CONCLUSION

Applicant submits that this paper provides a response for all pending claims. Any absence of a reply to a specific rejection, issue, or comment, or to any taking of "official notice" or reliance on "common sense", however, does not signify agreement with or concession of that rejection, issue, comment, taking of "official notice", or reliance on "common sense". In addition, because the arguments made above are not exhaustive, there may be reasons for patentability of any or all pending claims that have not been expressed.

In view of the amendments and arguments made herein, applicant submits that the application is in condition for allowance and requests early favorable action by the Examiner.

If the Examiner believes that a telephone conversation with the applicant's representative would expedite allowance of this application, the Examiner is cordially invited to call the undersigned at (508) 303-0932.

Respectfully submitted,

Date: March 2, 2009
Reg. No. 41,274

Tel. No.: (508) 303-0932
Fax No.: (508) 303-0005

/Michael A. Rodriguez/
Michael A. Rodriguez
Attorney for Applicant
Guerin & Rodriguez, LLP
5 Mount Royal Avenue
Marlborough, MA 01752

Attachment: Wikipedia definition of Xpath

XPath

From Wikipedia, the free encyclopedia

XPath (XML Path Language) is a language for selecting nodes from an XML document. In addition, XPath may be used to compute values (strings, numbers, or boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

XPath

Paradigm	query language
Appeared in	1999
Developer	W3C
Latest release	2.0/ January 23 2007
Major implementations	JavaScript, C#, Java
Influenced by	XSLT, XPointer
Influenced	XML Schema, XForms

Contents

History

The XPath language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria.[1] In popular use (though not in the official specification), an XPath expression is often referred to simply as *an XPath.*

Originally motivated by a desire to provide a common syntax and behavior model between XPointer and XSLT, subsets of the XPath query language are used in other W3C specifications such as XML Schema and XForms.

Versions

There are currently two versions in use.

XPath 1.0 became a Recommendation on 16 November 1999 and is widely implemented and used, either on its own (called via an API from languages such as Java, C# or JavaScript), or embedded in languages such as XSLT or XForms.

The current version of the language is XPath 2.0, which became a Recommendation on 23 January 2007. A number of implementations exist but are not as widely used as XPath 1.0. The XPath 2.0 language specification is much larger than XPath 1.0 and changes some of the fundamental concepts of the language such as the type system; the language specification is described in a separate article.

The most notable change is that XPath 2.0 has a much richer type system;[2] Every value is now a sequence (a single atomic value or node is regarded as a sequence of length one). XPath 1.0 node-sets are replaced by node sequences, which may be in any order.

To support richer type sets, XPath 2.0 offers a greatly expanded set of functions and operators.

XPath 2.0 is in fact a subset of XQuery 1.0. It offers a `for` expression which is cut-down version of the "FLWOR" expressions in XQuery. It is possible to describe the language by listing the parts of XQuery that it leaves out: the main examples are the query prolog, element and attribute constructors, the remainder of the "FLWOR" syntax, and the `typeswitch` expression.

See also

- XPath 1.0
- XPath 2.0

References

1. ^ Article on xpath in techsoftcomputing.com
2. ^ XPath 2.0 supports atomic types, defined as built-in types in XML Schema, and may also import user-defined types from a schema.[1]

External links

- XPath 1.0 specification
- XPath 2.0 specification
- What's New in XPath 2.0

Retrieved from "http://en.wikipedia.org/wiki/XPath"
Categories: XML | XML data access
Hidden categories: All articles to be merged | Articles to be merged since September 2008